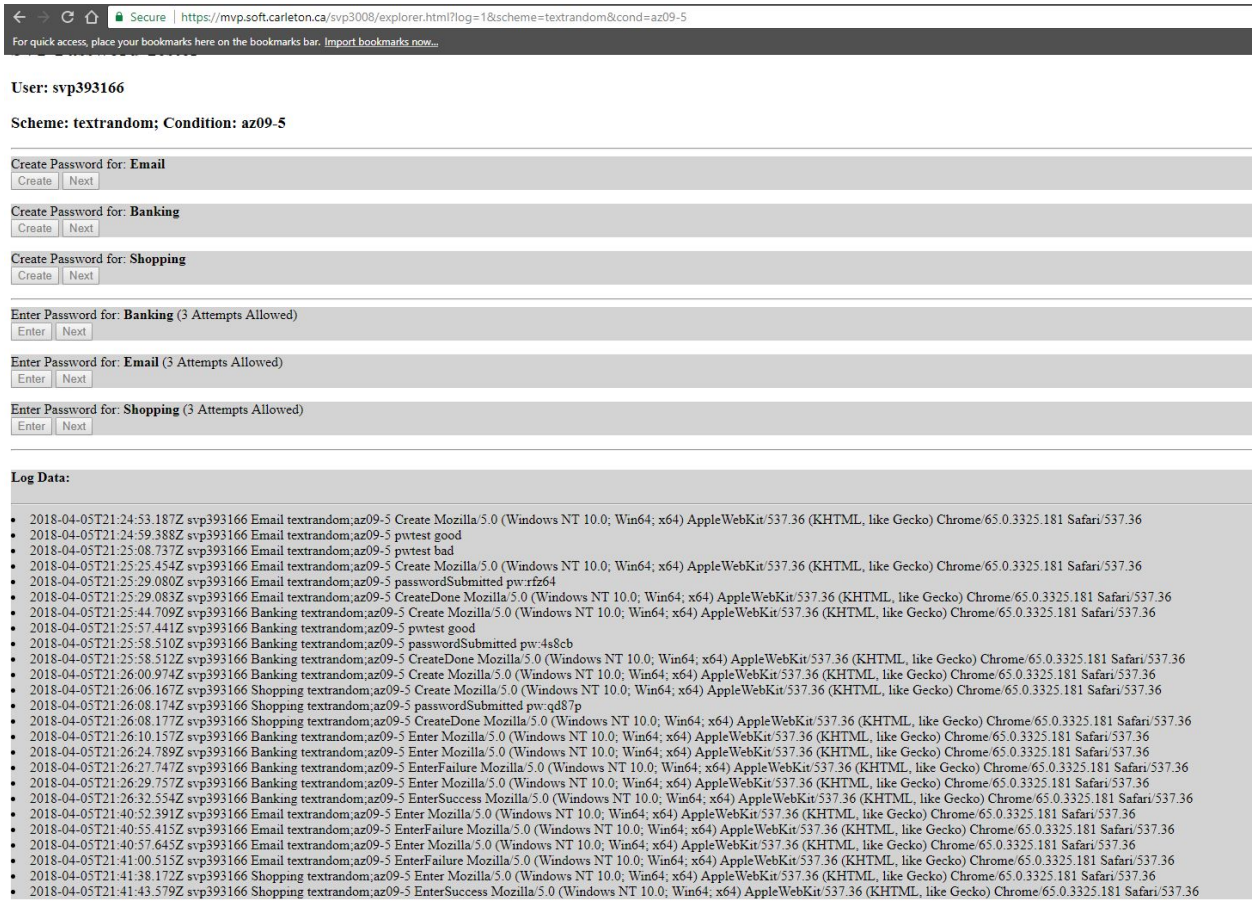
COMP3008 Project 2
Ryan Lau 100972349

Part 1. 1.

Text21:



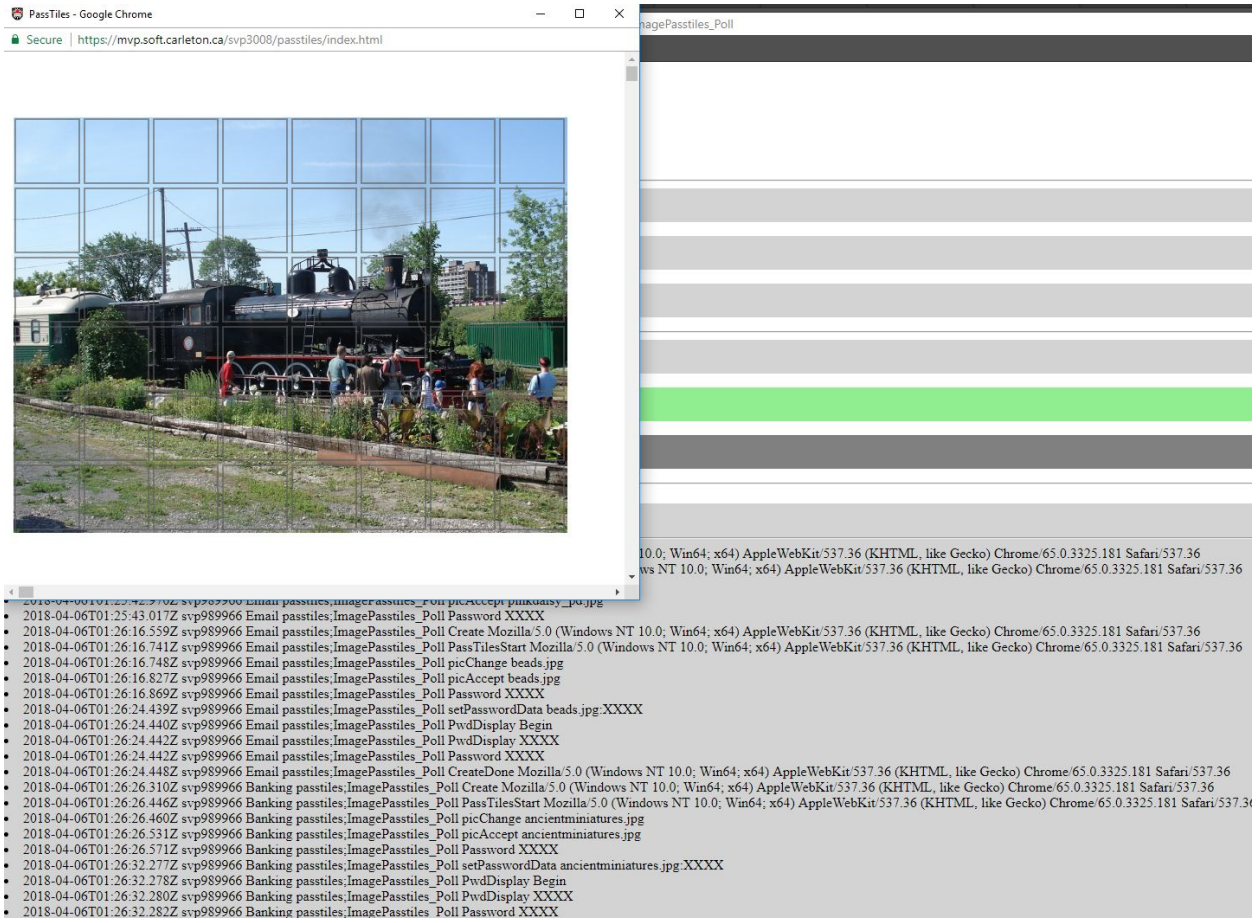
Advantages**:**

This scheme was simple, easy to use and memorable. Remembering 4 random letters and a digit was much easier than the other password scheme. It is also relatively secure, with 36^5 possible combinations.

Disadvantages:

It can be harder to memorize this than a user-made password, since users can make their passwords contain words, phrases or anything that they find easy to memorize or prefer. Other than that,

Imagept21:



Advantages:

It can be easier to memorize depending on the picture. In this case, I remembered that the person in the red shirt was one square, acting as a point of reference.

Disadvantages:

This scheme seemed much more difficult to memorize. The picture helps to remember which square exactly is the right ones, but also threw me off. The picture can also be detrimental, if the picture is hard to see, has many patterns, has nothing outstanding to spot the right squares, is hard to spot differences, etc.

Part 1. 2.

For my log data processing software, I used java, reading in the appropriate file and splitting them according to their lines and commas. For the output, the userid and password scheme were simply the ones read in, in columns two and five, respectively. For the number of logins, I had to check when a login was successful or failed (which was found in column six), and

backtrack the log to see when the user started the attempts to login. There are two separate values for this, one for the successful logins, and one for the number of failed logins. The total number of logins was just the two added together up to that point. Finally, for the time taken between entering passwords, I checked the time found in column one in order to compare the times of two separate login attempts. Subtracting the times together gave me the exactly what I needed.

Pseudocode

Read in file
Split the file according to lines
Split lines according to the commas
Loop through all lines

Output UserId
Output PasswordScheme

If Login was successful or failed
    Check the number of login attempts by going back lines
    Check the time it took in between attempts

Output number of login attempts
Output time

Note:
The format of the output is :

String userid, String scheme, int timeElapsed, int successFail, int Count, int FailCount, intSuccessCount

Userid represents the id of the user logging in.
Scheme represents the password scheme.
timeElapsed represents the time since the user was prompted for the password, in seconds.
successFail represents whether or not the login was a pass or fail. 1 for success 0 for fail.
The last 3 are just a counter for the amount of attempted logins.

Part 1. 3.

Text21

```
> aggregate(data1[,5:7],list(data1$User),mean)
   Group.1 Count SuccessCount FailCount
1   ast103   9.0     8.176471 0.8235294
2   ast104  13.5     6.000000 7.5000000
3   ast105   8.0     5.733333 2.2666667
4   ast107  11.0     7.904762 3.0952381
5   ast108  10.0    10.000000 0.0000000
6   ast111  10.5     3.900000 6.6000000
7   ast112   8.0     8.000000 0.0000000
8   ast114  11.5     6.227273 5.2727273
9   ast115   8.0     8.000000 0.0000000
10  ast116   8.0     8.000000 0.0000000
11  ast118   8.5     7.562500 0.9375000
12  ast125   9.5     7.555556 1.9444444
13  ast131  11.0     6.619048 4.3809524
14  ast133   5.0     5.000000 0.0000000
15  ast134   8.0     8.000000 0.0000000
16  ast135   2.5     1.500000 1.0000000
17  ast136   8.0     8.000000 0.0000000
18  ast138   8.5     8.500000 0.0000000
```

```
> aggregate(data1[,5:7],list(data1$User),median)
   Group.1 Count SuccessCount FailCount
1   ast103   9.0          8.0         1
2   ast104  13.5          4.5         9
3   ast105   8.0          5.0         3
4   ast107  11.0          8.0         3
5   ast108  10.0         10.0         0
6   ast111  10.5          2.5         8
7   ast112   8.0          8.0         0
8   ast114  11.5          4.5         7
9   ast115   8.0          8.0         0
10  ast116   8.0          8.0         0
11  ast118   8.5          7.5         1
12  ast125   9.5          7.5         2
13  ast131  11.0          5.0         6
14  ast133   5.0          5.0         0
15  ast134   8.0          8.0         0
16  ast135   2.5          1.5         1
17  ast136   8.0          8.0         0
18  ast138   8.5          8.5         0
```

```
> aggregate(data1[,5:7],list(data1$User),sd)
   Group.1    Count SuccessCount FailCount
1   ast103 5.049752     4.798897 0.3929526
2   ast104 7.648529     5.067544 3.1272992
3   ast105 4.472136     3.575046 1.1629192
4   ast107 6.204837     4.265029 1.9976176
5   ast108 5.627314     5.627314 0.0000000
6   ast111 5.916080     4.266146 2.3033157
7   ast112 4.472136     4.472136 0.0000000
8   ast114 6.493587     4.576668 2.3131471
9   ast115 4.472136     4.472136 0.0000000
10  ast116 4.472136     4.472136 0.0000000
11  ast118 4.760952     4.661455 0.2500000
12  ast125 5.338539     4.792423 0.6391375
13  ast131 6.204837     4.364358 2.3340135
14  ast133 2.738613     2.738613 0.0000000
15  ast134 4.472136     4.472136 0.0000000
16  ast135 1.290994     1.290994 0.0000000
17  ast136 4.472136     4.472136 0.0000000
18  ast138 4.760952     4.760952 0.0000000
```

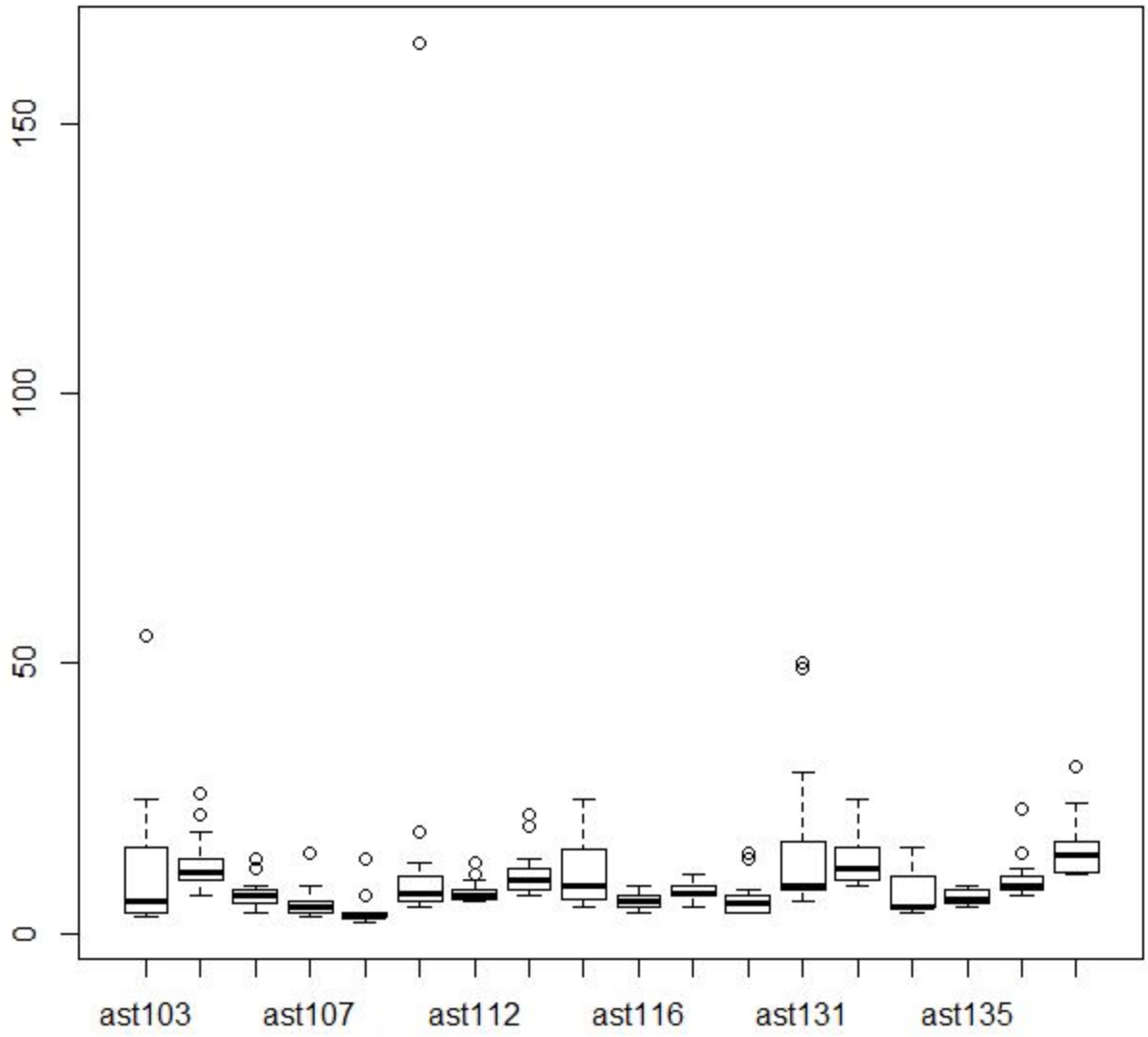
Text21 descriptive statistics for number of logins. Count represents total logins while SuccessCount represents successes and FailCount represents fails. The first picture shows mean, the second shows median and the third is the standard deviation.

```
> aggregate(data1[,3:4],list(data1$User),mean)
   Group.1      Time     Login
1   ast103 12.529412 0.9411765
2   ast104 12.730769 0.6153846
3   ast105  7.333333 0.8000000
4   ast107  5.523810 0.7142857
5   ast108  4.210526 1.0000000
6   ast111 16.200000 0.6000000
7   ast112  7.733333 1.0000000
8   ast114 10.727273 0.6818182
9   ast115 11.200000 1.0000000
10  ast116  5.933333 1.0000000
11  ast118  7.875000 0.9375000
12  ast125  6.333333 0.8333333
13  ast131 15.047619 0.7142857
14  ast133 14.555556 1.0000000
15  ast134  7.533333 1.0000000
16  ast135  6.750000 0.7500000
17  ast136 10.333333 1.0000000
18  ast138 15.687500 1.0000000
```

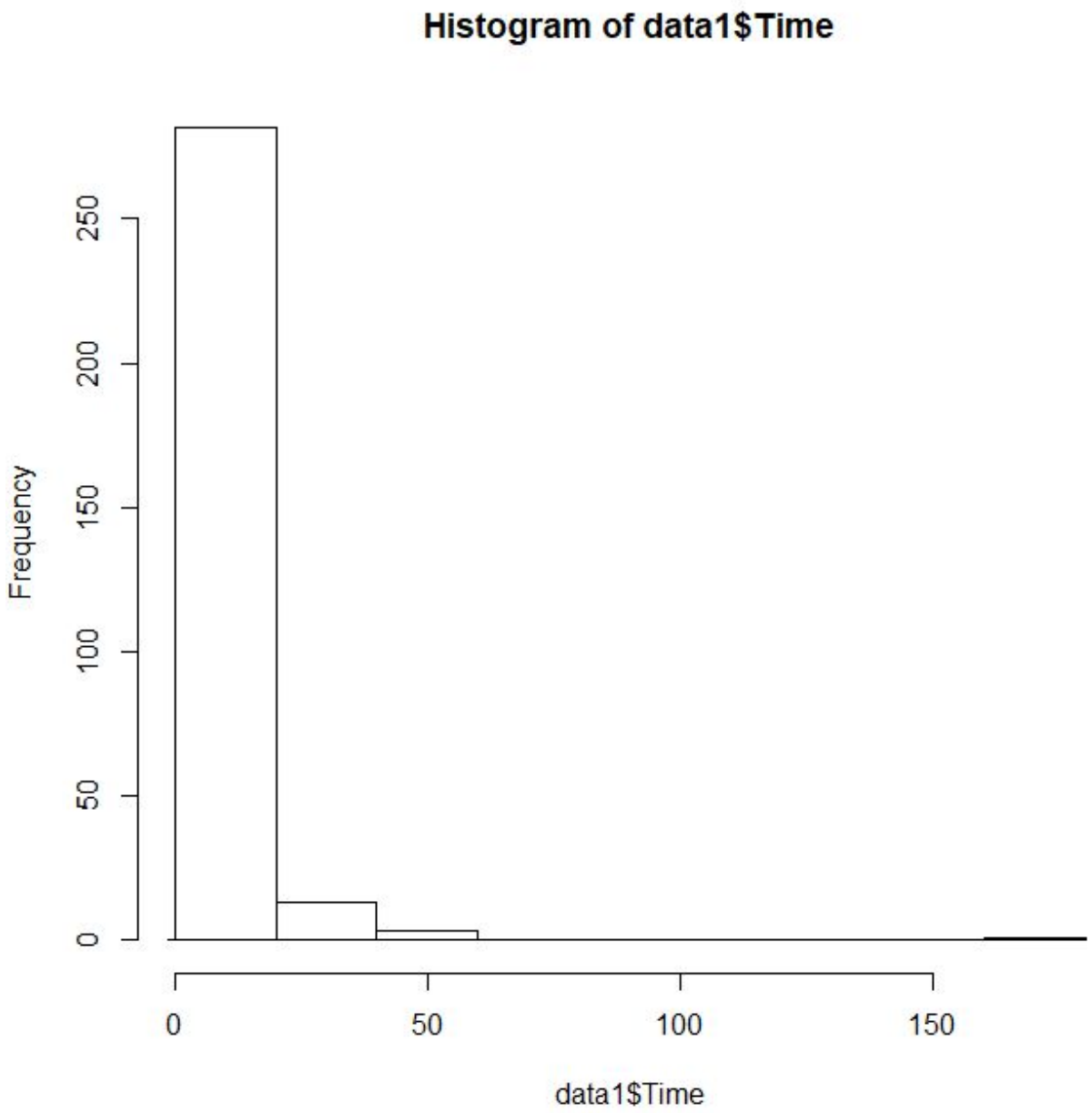
```
> aggregate(data1[,3:4],list(data1$User),median)
   Group.1 Time Login
1   ast103  6.0     1
2   ast104 11.5     1
3   ast105  7.0     1
4   ast107  5.0     1
5   ast108  3.0     1
6   ast111  7.5     1
7   ast112  7.0     1
8   ast114 10.0     1
9   ast115  9.0     1
10  ast116  6.0     1
11  ast118  7.5     1
12  ast125  5.5     1
13  ast131  9.0     1
14  ast133 12.0     1
15  ast134  5.0     1
16  ast135  6.5     1
17  ast136  9.0     1
18  ast138 14.5     1
```

```
> aggregate(data1[,3:4],list(data1$User),sd)
   Group.1       Time     Login
1   ast103 13.200936 0.2425356
2   ast104  4.459217 0.4961389
3   ast105  2.742956 0.4140393
4   ast107  2.676174 0.4629100
5   ast108  2.699361 0.0000000
6   ast111 35.186122 0.5026247
7   ast112  2.086236 0.0000000
8   ast114  4.014045 0.4767313
9   ast115  6.014269 0.0000000
10  ast116  1.486447 0.0000000
11  ast118  1.707825 0.2500000
12  ast125  3.217599 0.3834825
13  ast131 13.086161 0.4629100
14  ast133  5.703313 0.0000000
15  ast134  3.700708 0.0000000
16  ast135  1.707825 0.5000000
17  ast136  4.064949 0.0000000
18  ast138  5.338149 0.0000000
```

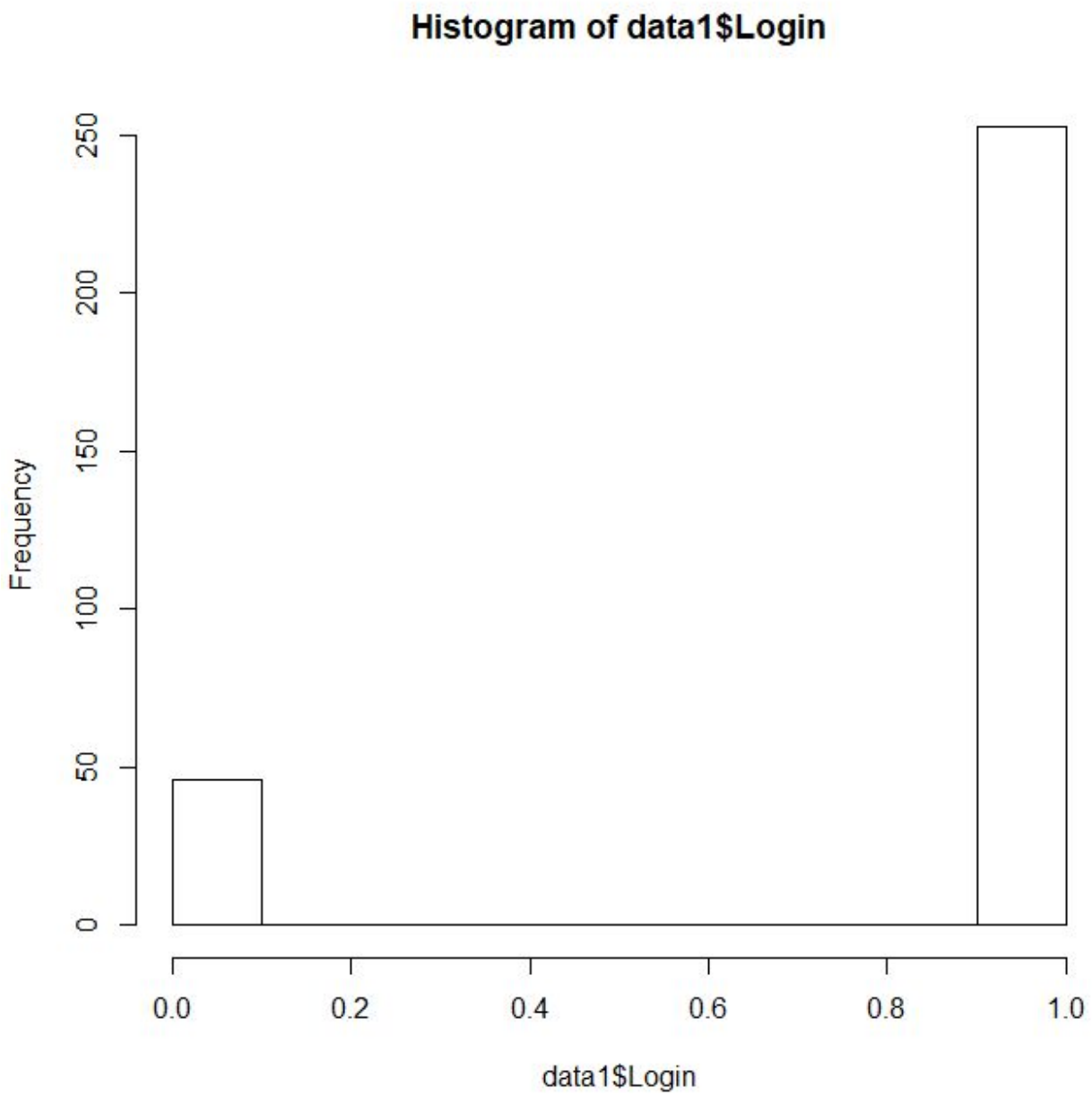
Text21 descriptive statistics for time between logins. Login value represents success or fail, 1 would be a success and 0 a fail, so a mean of 0.5 would mean the user passed half of the login prompts.



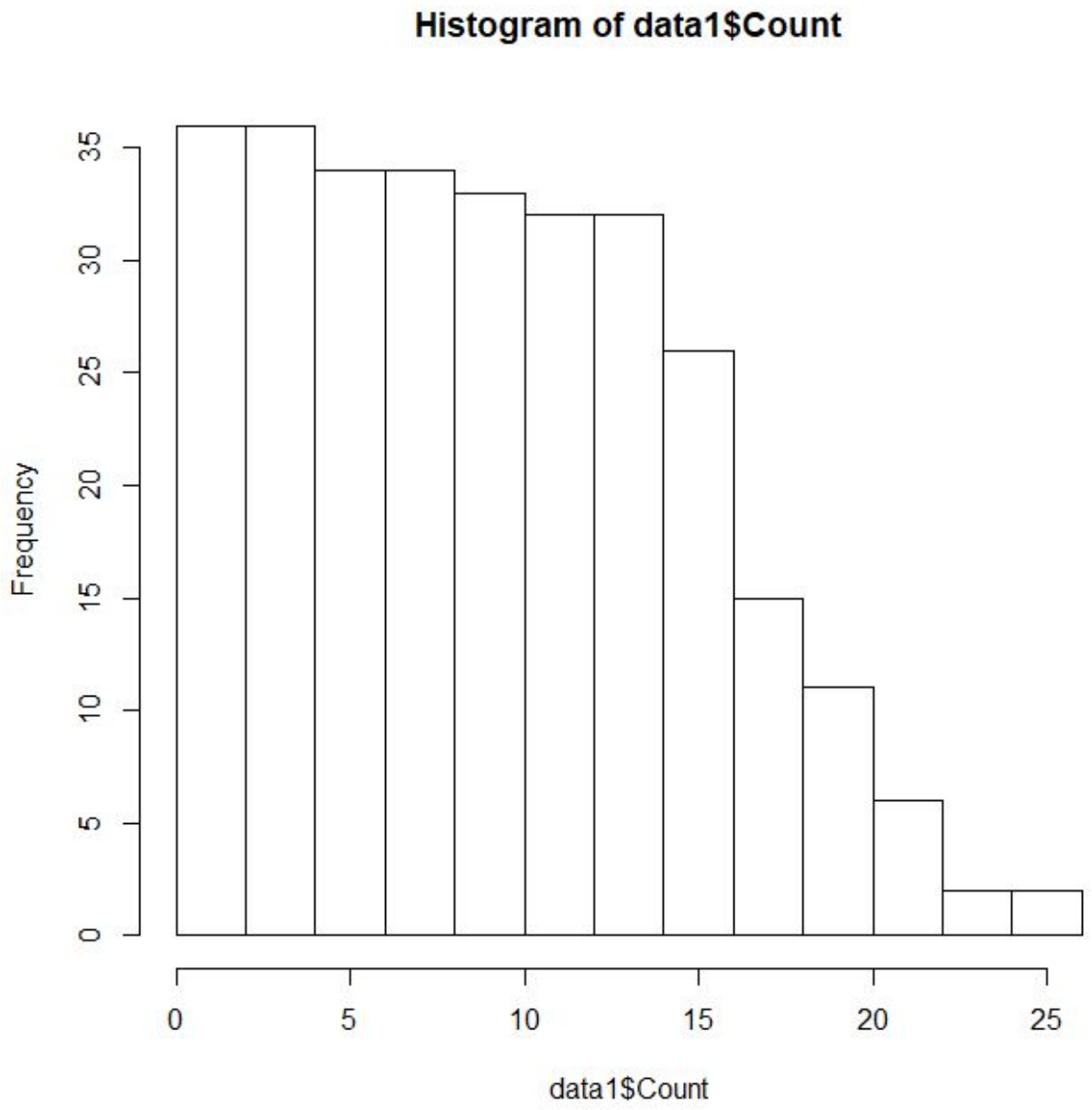
Text21's boxplot for the time it took to login per user. There is are a few outliers here and there.



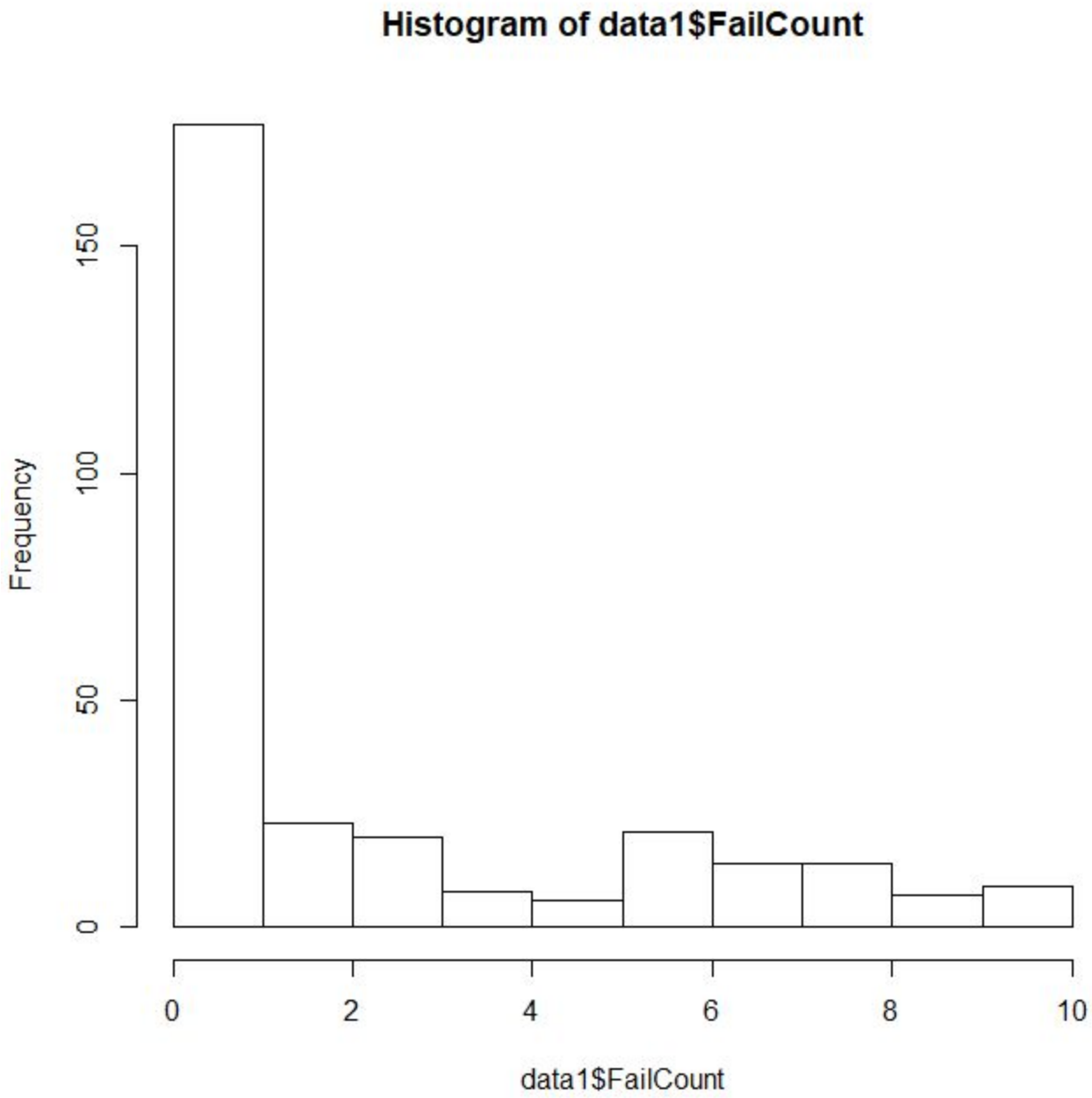
The histogram for the time to login. As expected, the majority of response time was less than 1 minute, while the longest one was above 2 minutes.



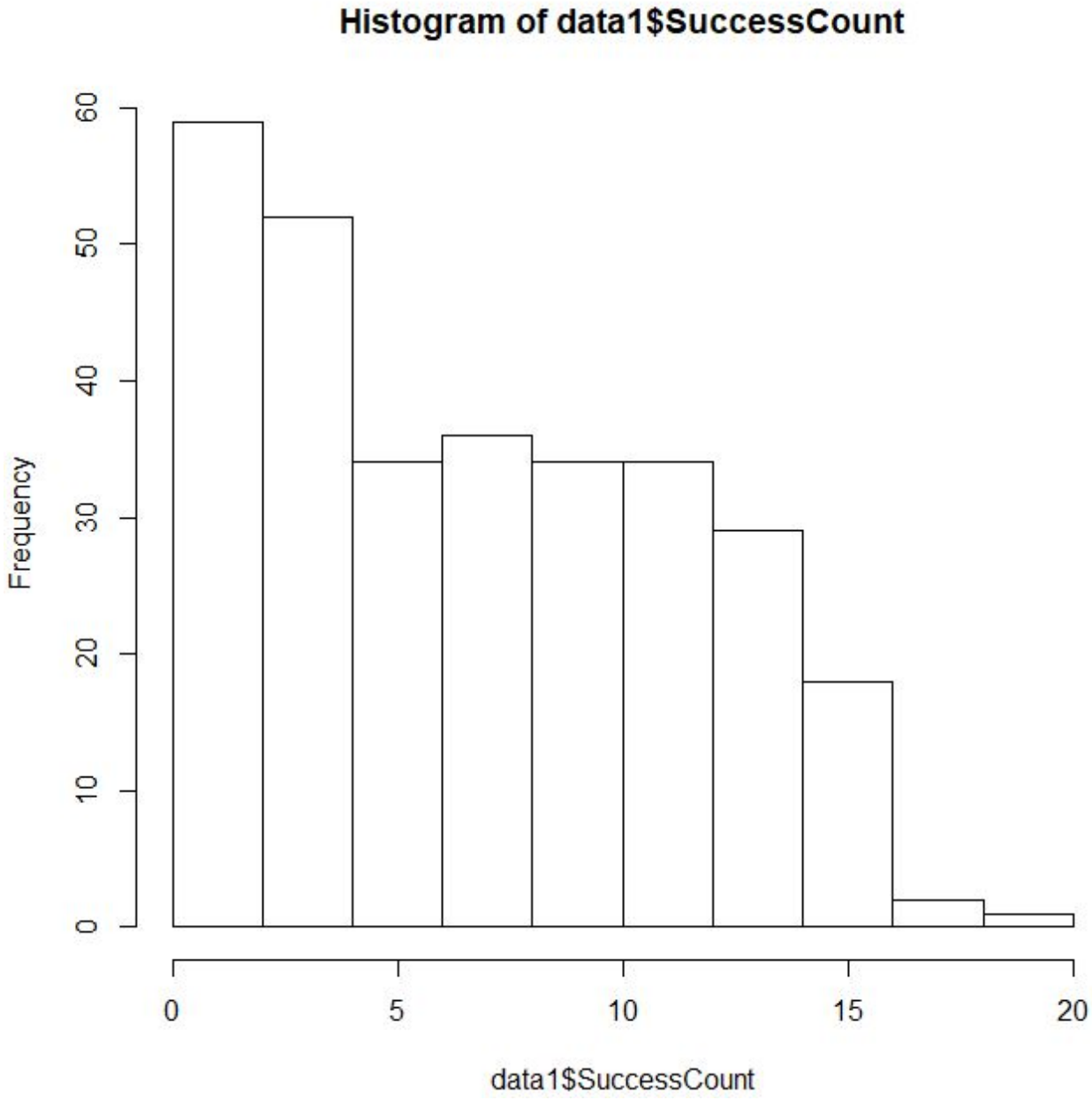
The histogram for the amount of login passes or fails. 1 represents success and a 0 represents a fail. This graph show ~250 successes and ~50 fails, representing an approximate 84% pass rate.



Histogram for the amount of logins per user. As the amount of logins increase, the frequency decreases.



Histogram for the amount of failed logins. The frequency for 1 fail represents most of the fail count.



Histogram for the amount of successes. As the amount of successes increase, the frequency decreases.

Imagept21

```
> aggregate(data2[,5:7],list(data2$User),mean)
   Group.1 Count SuccessCount FailCount
1   ipt101  16.5     5.875000 10.625000
2   ipt104   6.5     6.500000  0.000000
3   ipt105  11.5     7.636364  3.863636
4   ipt106  11.0     6.666667  4.333333
5   ipt109   8.0     8.000000  0.000000
6   ipt110  10.0     6.473684  3.526316
7   ipt113  15.0     8.137931  6.862069
8   ipt119   9.5     8.000000  1.500000
9   ipt131   8.0     6.666667  1.333333
10  ipt133   8.5     7.500000  1.000000
11  ipt134  10.5     7.300000  3.200000
12  ipt136   9.5     7.333333  2.166667
13  ipt137  10.5     7.300000  3.200000
14  ipt143   9.5     7.777778  1.722222
15  ipt145   8.5     7.812500  0.687500
```

```
> aggregate(data2[,5:7],list(data2$User),median)
   Group.1 Count SuccessCount FailCount
1   ipt101  16.5          4.0      12.5
2   ipt104   6.5          6.5       0.0
3   ipt105  11.5          7.5       4.0
4   ipt106  11.0          6.0       5.0
5   ipt109   8.0          8.0       0.0
6   ipt110  10.0          6.0       4.0
7   ipt113  15.0          7.0       8.0
8   ipt119   9.5          7.5       2.0
9   ipt131   8.0          7.0       1.0
10  ipt133   8.5          7.5       1.0
11  ipt134  10.5          6.5       4.0
12  ipt136   9.5          6.5       3.0
13  ipt137  10.5          6.5       4.0
14  ipt143   9.5          7.5       2.0
15  ipt145   8.5          7.5       1.0
```

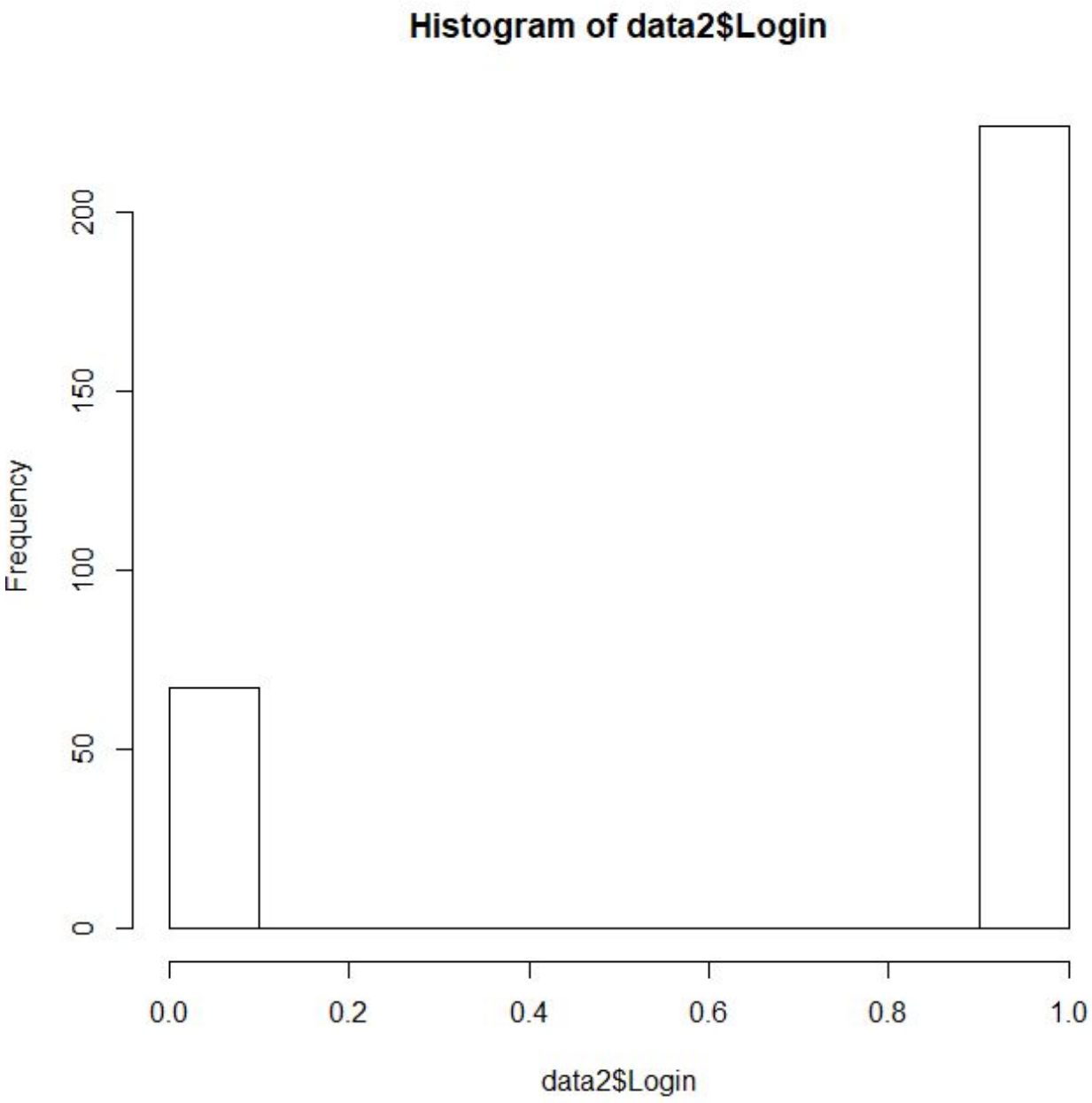
```
> aggregate(data2[,5:7],list(data2$User),sd)
   Group.1    Count SuccessCount FailCount
1   ipt101 9.380832     4.404909 5.4521378
2   ipt104 3.605551     3.605551 0.0000000
3   ipt105 6.493587     4.370305 2.2317077
4   ipt106 6.204837     4.715224 1.8257419
5   ipt109 4.472136     4.472136 0.0000000
6   ipt110 5.627314     5.015182 0.8411910
7   ipt113 8.514693     4.525516 4.1120393
8   ipt119 5.338539     4.702940 0.8574929
9   ipt131 4.472136     3.265986 1.2909944
10  ipt133 4.760952     4.760952 0.0000000
11  ipt134 5.916080     4.105196 2.0157276
12  ipt136 5.338539     4.338609 1.2947859
13  ipt137 5.916080     4.900054 1.3992479
14  ipt143 5.338539     4.634342 0.8947925
15  ipt145 4.760952     4.385107 0.4787136
```

```
> aggregate(data2[,3:4],list(data2$User),mean)
   Group.1       Time     Login
1   ipt101  19.750000 0.5000000
2   ipt104  21.916667 1.0000000
3   ipt105  10.909091 0.6818182
4   ipt106  17.904762 0.7142857
5   ipt109  23.133333 1.0000000
6   ipt110  17.842105 0.7894737
7   ipt113   9.827586 0.5862069
8   ipt119  13.555556 0.8888889
9   ipt131  39.066667 0.8000000
10  ipt133 -58.000000 0.9375000
11  ipt134  27.400000 0.7500000
12  ipt136  13.277778 0.8333333
13  ipt137  20.050000 0.8000000
14  ipt143  22.555556 0.8333333
15  ipt145  25.312500 0.9375000
```

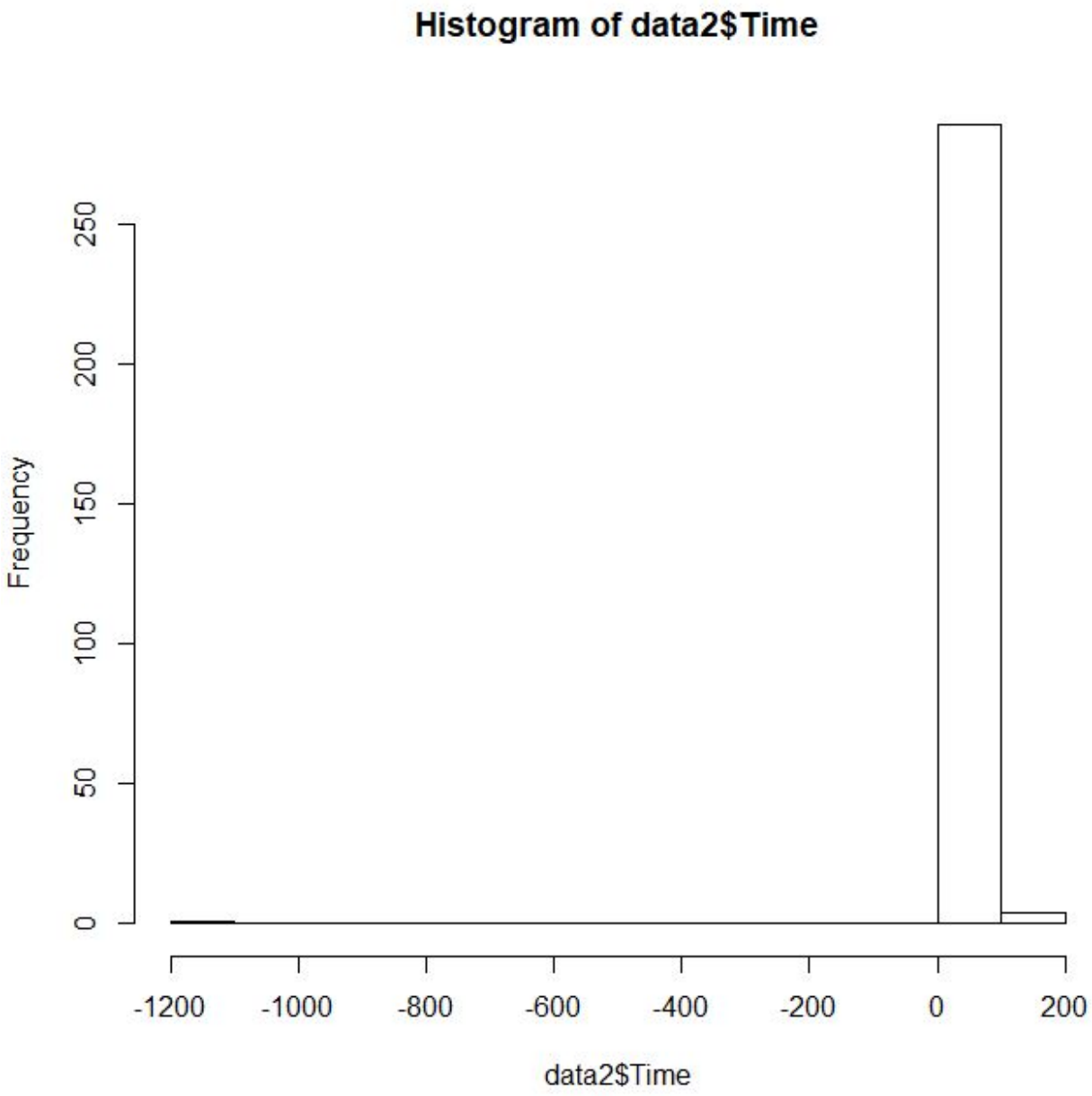
```
> aggregate(data2[,3:4],list(data2$User),median)
   Group.1 Time Login
1   ipt101 14.5   0.5
2   ipt104 21.5   1.0
3   ipt105  7.5   1.0
4   ipt106 11.0   1.0
5   ipt109 17.0   1.0
6   ipt110 11.0   1.0
7   ipt113  6.0   1.0
8   ipt119 10.5   1.0
9   ipt131 35.0   1.0
10  ipt133 14.5   1.0
11  ipt134 21.5   1.0
12  ipt136 12.5   1.0
13  ipt137 18.5   1.0
14  ipt143 10.0   1.0
15  ipt145 24.0   1.0
```

```
> aggregate(data2[,3:4],list(data2$User),sd)
   Group.1       Time     Login
1   ipt101  16.962958 0.5080005
2   ipt104   6.625822 0.0000000
3   ipt105   7.334907 0.4767313
4   ipt106  30.572708 0.4629100
5   ipt109  12.631405 0.0000000
6   ipt110  24.044549 0.4188539
7   ipt113  12.409872 0.5012300
8   ipt119  12.598890 0.3233808
9   ipt131  14.562509 0.4140393
10  ipt133 288.830285 0.2500000
11  ipt134  26.230557 0.4442617
12  ipt136   3.937834 0.3834825
13  ipt137   7.163320 0.4103913
14  ipt143  39.966489 0.3834825
15  ipt145   6.117938 0.2500000
```

The mean, median and standard variation values of login times, and successful logins, as well as the login count (success and fails).

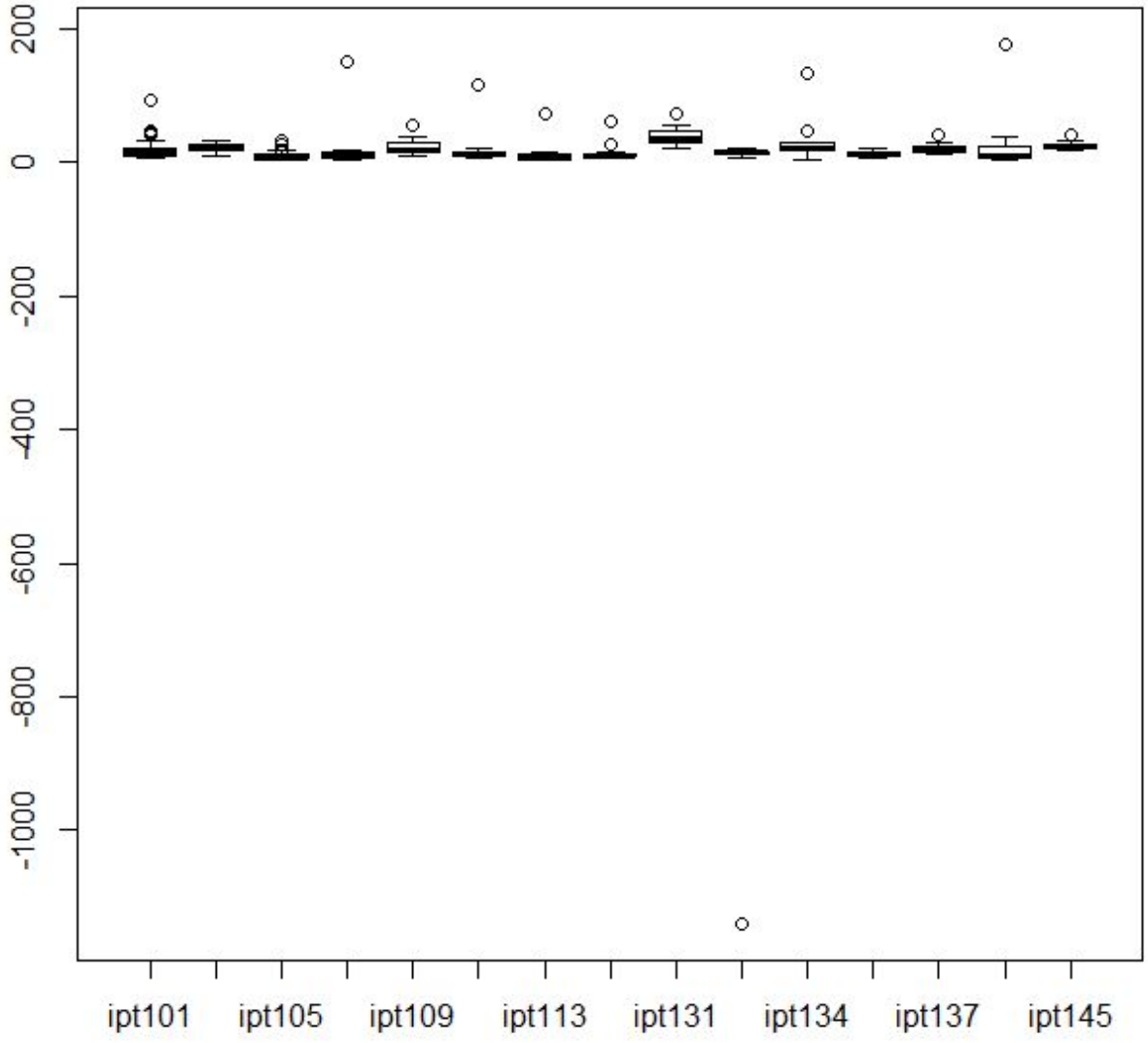


Histogram for the amount of logins. 1.0 represents success while 0.0 a fail. Interestingly, the login pass rate is lower than that of Text21's, which had ~250 passes and ~50 fails. This password scheme boasts ~240 passes and ~60 fails, giving it a 80% pass rate.

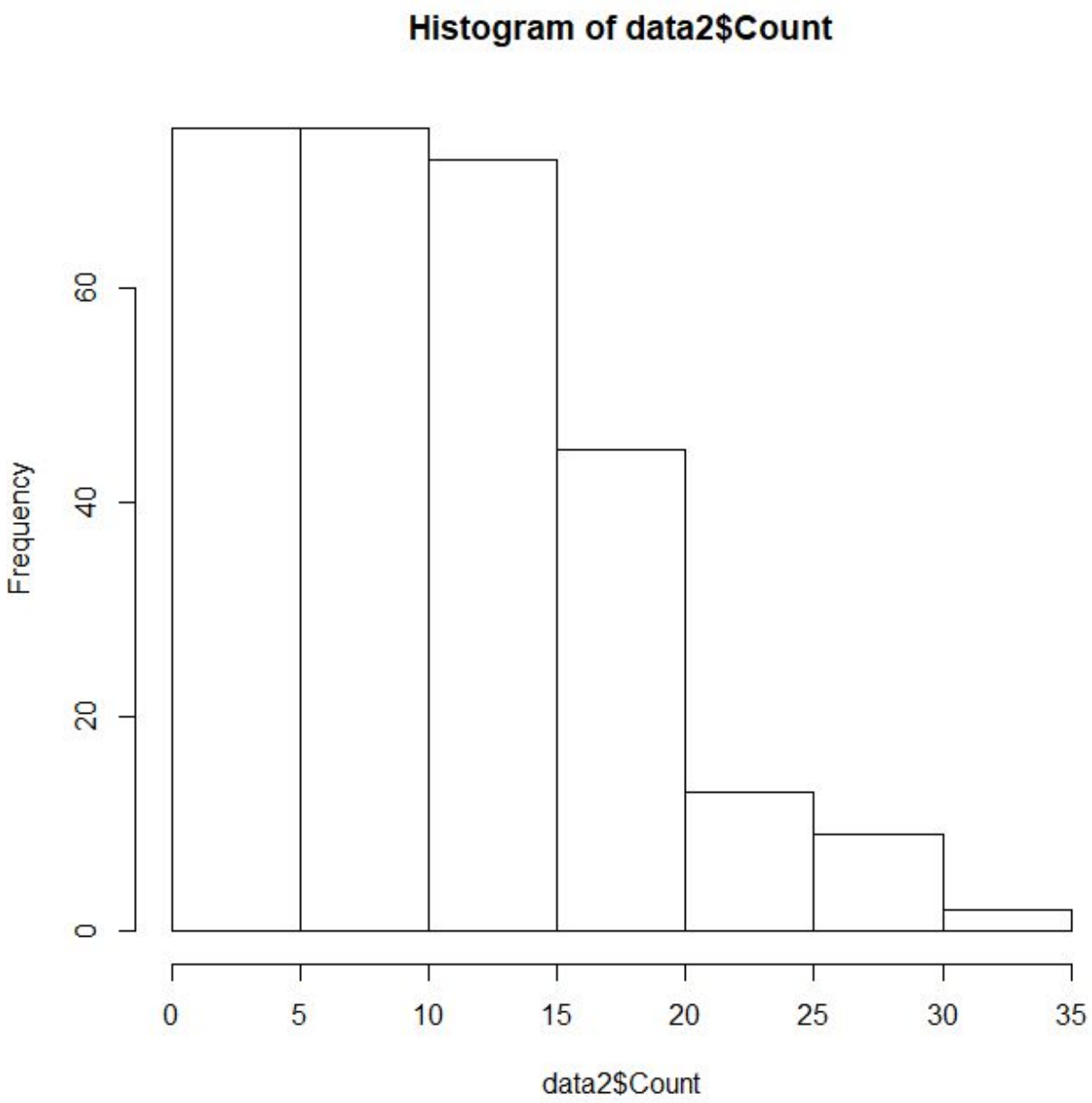


The histogram for the time to login.

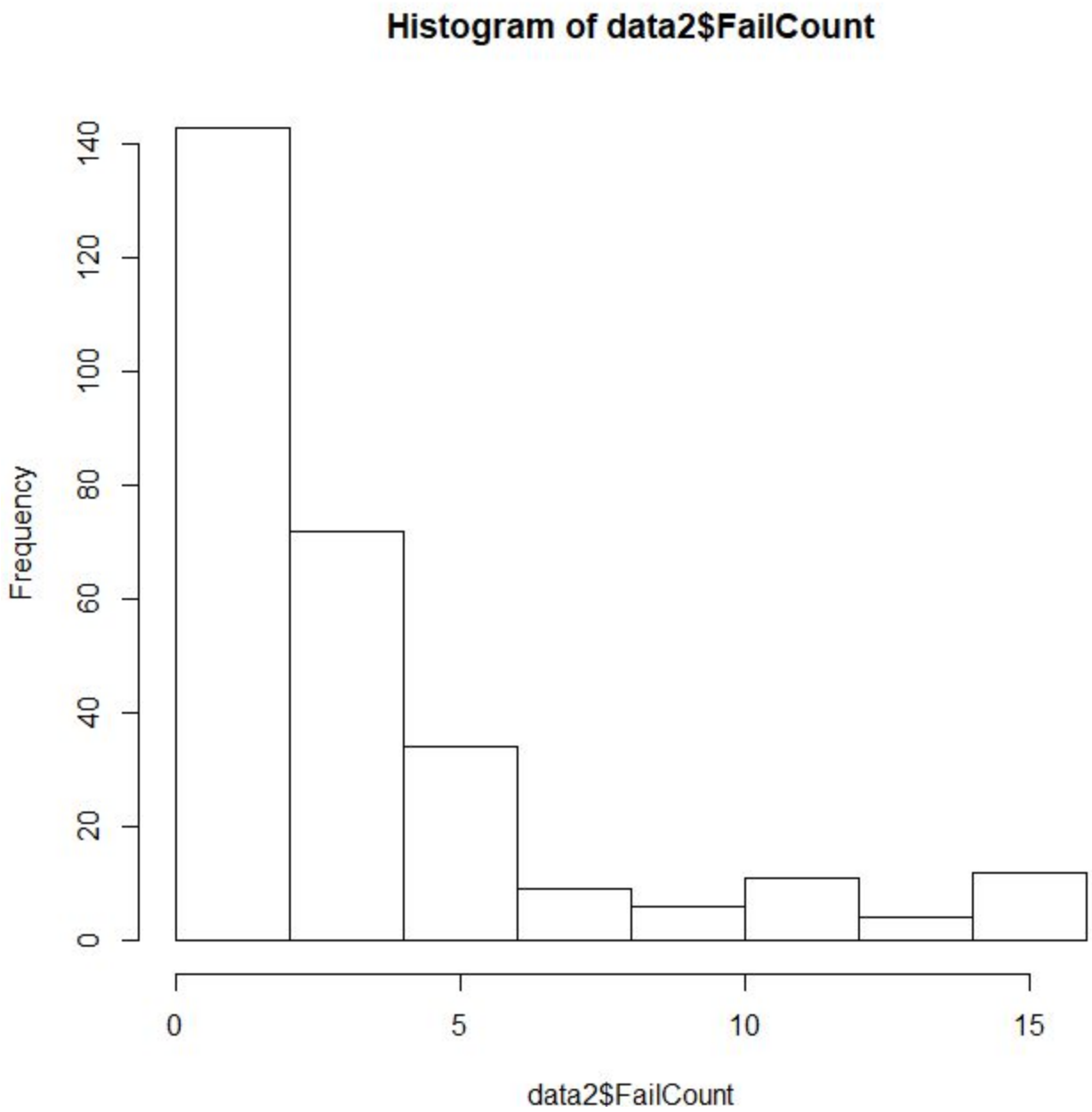
.



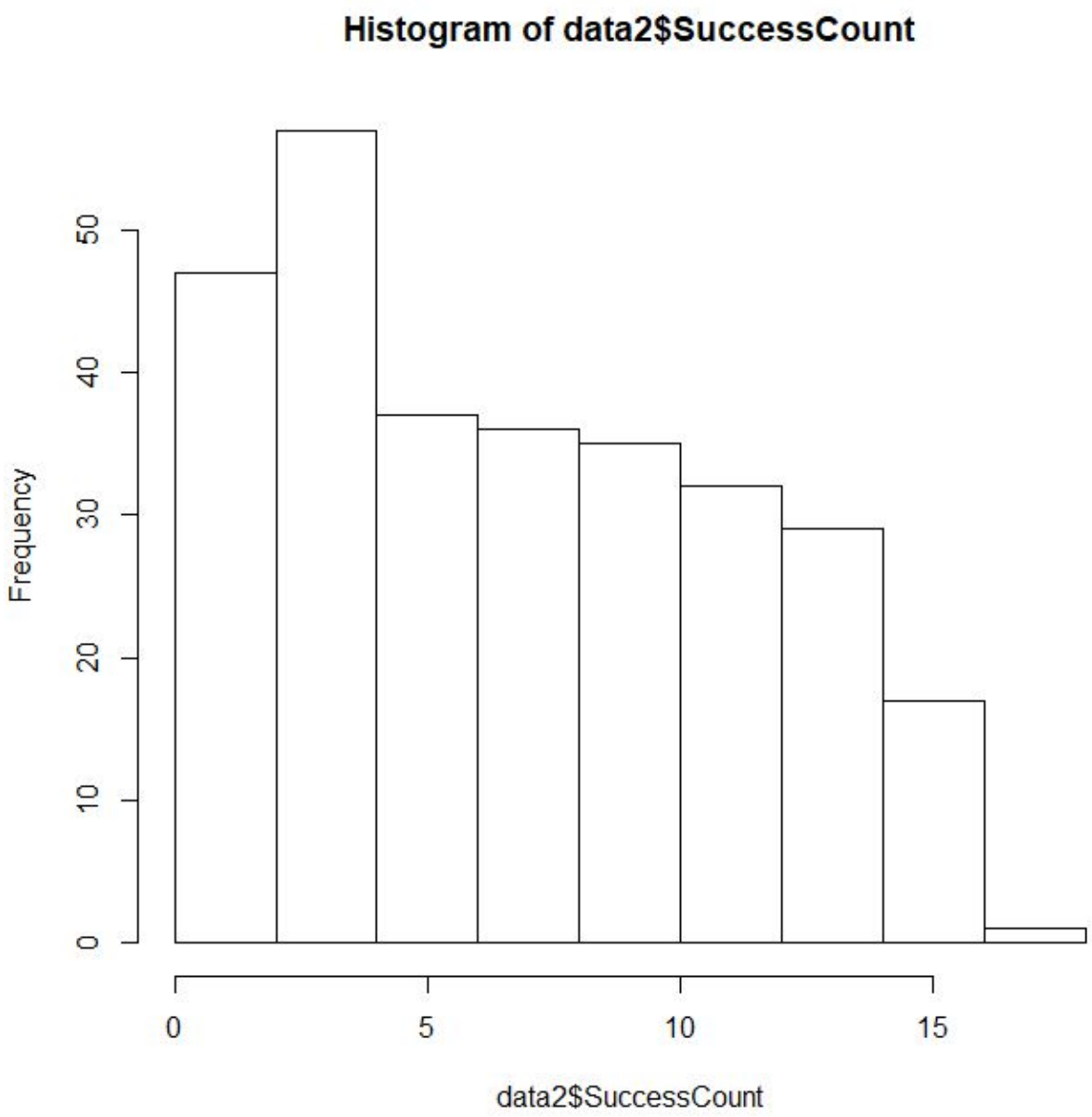
The boxplot for the login times per user. The times seem relatively similar to that of Text21's.



Histogram for the amount of logins per user. As the amount of logins increase, the frequency decreases. This one looks relatively similar to that of Text21's.



Histogram for the amount of failed logins for Imagept21. The frequency for 1 fail represents most of the fail count. The amount of fails near 0~5 are much higher than those of text21.



The amount of successful logins for Imagept21. Once again, similar to Text21's, as the amount of successes increase, the frequency decreases.

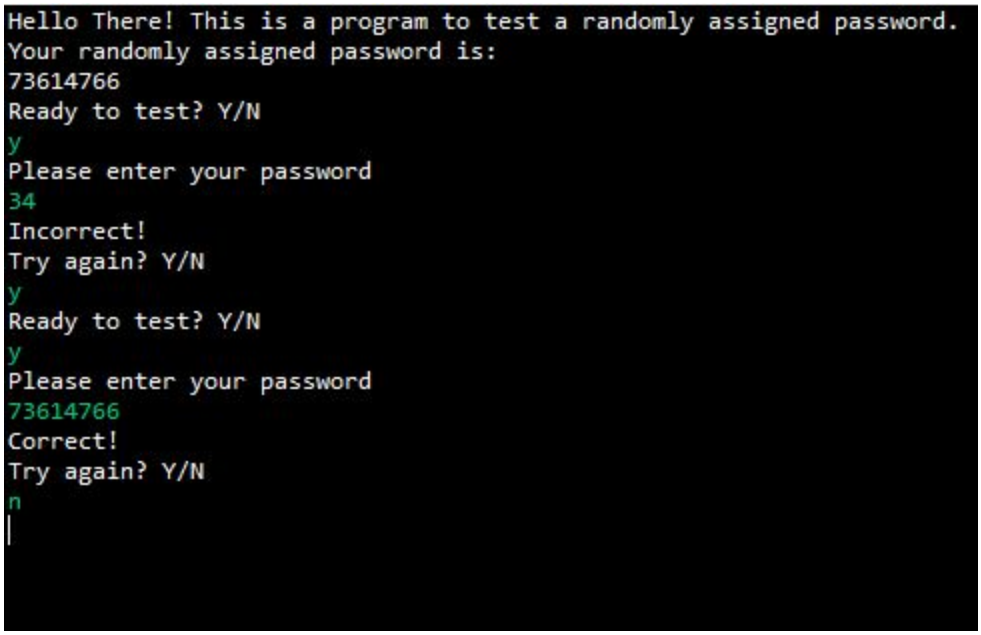
To conclude, many of the features in the graphs and tables of the two schemes are quite similar. They both fall off for count, and the login times seem to be more or less the same. However, the login success rate for Text21 was 84%, slightly higher than that Imagept21's 80%. From the results, if I had to choose the better scheme, it would be Text21, boasting less time for password entry time, and lower amount of failed logins.

Part 2. 1.

Example Password: 34512337

For this new authentication scheme, I chose to assign 8 random numbers from 1-7. Using this method, assuming each of the 8 numbers can only be 1,2,3,4,5,6 or 7 there are then 8^7 possibilities, or 2,097,152 possibilities, the exact same possibility to 2^{21} or 2,097,152 suggested possibilities. Although it could be harder to memorize 8 numbers, it should be relatively easy after getting used to it. It also doesn't help that there is nothing to help with the user's memory, since there is only numbers and no letters for pattern/word recognition. However, it is straightforward and easier for users to understand, while being decently secure. I decided to go with this scheme over something like letters, words and images, due to its ease of use. I also believe that text passwords are much easier to use as well as implement. Overall, I think this scheme is simple but it could have potential.

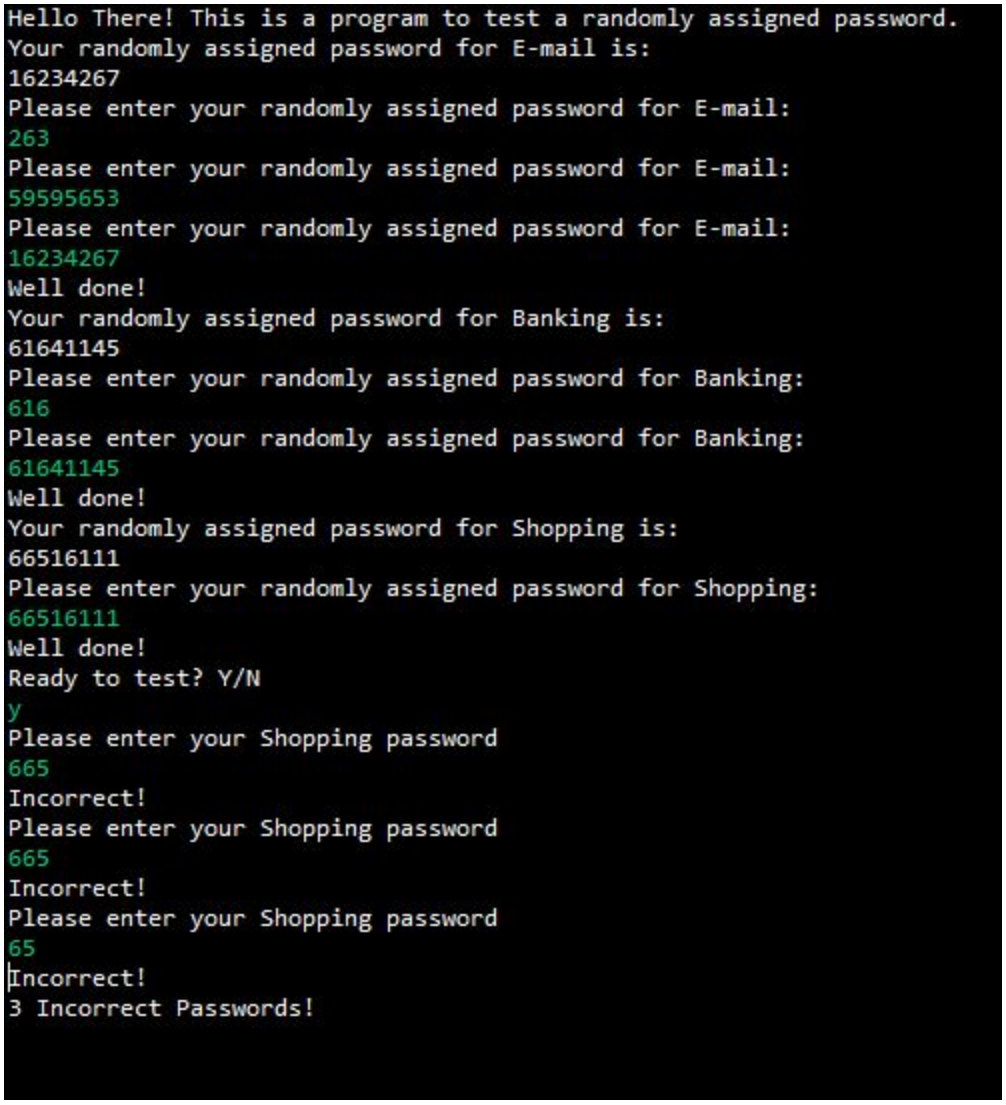
Part 2. 2.



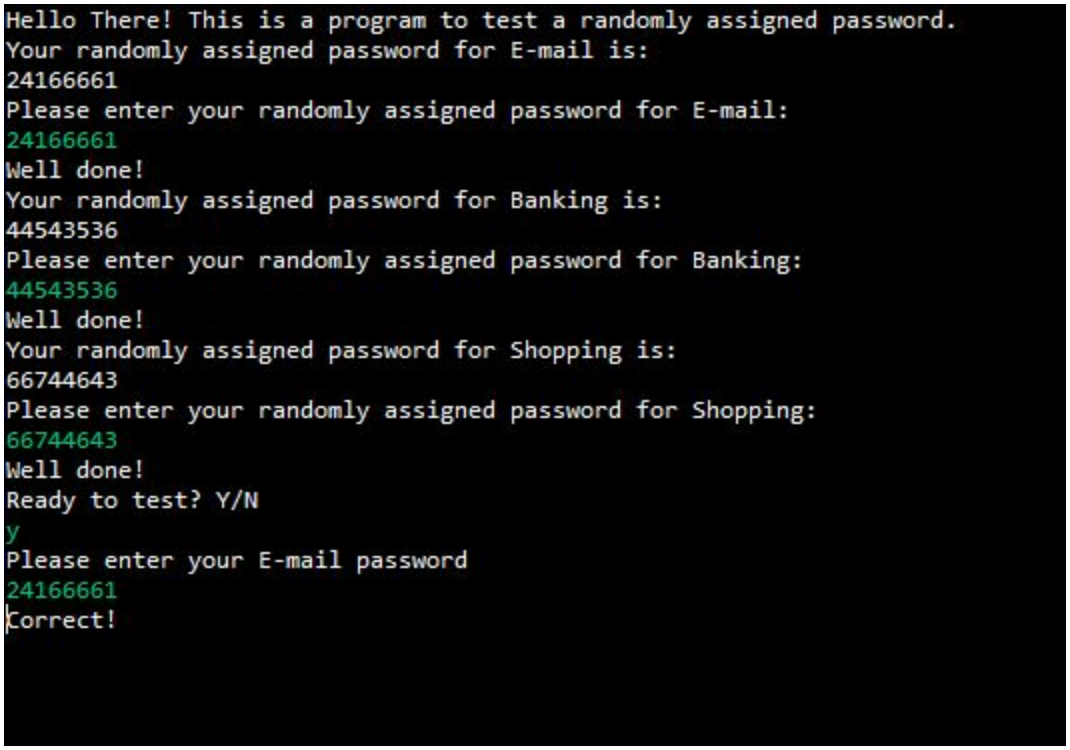
Once again I used java here. The program is relatively simple, it randomly assigns 8 numbers from 1-7 and tests the user to see whether or not they can get it correctly. I've added a function in case the user incorrectly puts in their password. In this case, they can try again, and hopefully

get it correctly, or continue to try. The password generation is done by assigning a random number 8 times and then concatenating them into a string. This is just the basic password generation and entry.

Part 2.3.



In this part, I created three random passwords for each of e-mail, banking and shopping. The program will ask the user to re-enter the password for confirmation, before jumping to the next password. It will do so indefinitely until the user enters the password correctly. After confirmation of all three passwords, the user can proceed to test their password. Getting the password wrong three times will terminate the program. The data will also be logged into a file called LogData.txt.




```
2018,04,05 .23:48:57, E-mail password is 24166661
2018,04,05 .23:48:57, Banking password is 44543536
2018,04,05 .23:48:57, Shopping password is 66744643
2018,04,05 .23:48:57, Correct password for E-mail
```

An example of the log data along with the program running. The left side shows the date, while the left side shows the log. The log data logs almost every user controlled action, including password generation, entry and failure after 3 incorrect password entries. This log is something like the input file of part1, but the end goal is the output file.

```
2018-04-06 13:01:04,create,start
2018-04-06 13:01:08,create,start
2018-04-06 13:01:13,create,start
2018-04-06 13:01:18,enter,start
2018-04-06 13:01:18,12,login,failure,1,0,1
2018-04-06 13:01:18,enter,start
2018-04-06 13:01:18,16,login,success,2,1,1
```

The same log in the given csv format. Create, start are the entries where the password is generated, while enter,start is when the user is prompted to enter their password. Login is when the user submits the password and is either a success or a fail. The number before login in this case represents the amount of seconds that have elapsed since the prompt to enter the password. The last 3 entries are the number of logins, number of successful logins, and number of failed logins.

```
lop,P2,8,0,1,0,1
lop,P2,13,0,2,0,2
lop,P2,24,1,3,1,2
```

The log in the final format, removing the dates and only logging actual logins. First two entries are user and scheme. The third one represents the time in seconds to login, the fourth whether or not the login passed (1 for pass, 0 for fail), and the last 3 are the amount of login counts.

Part 2.4.

1.I prefer this password scheme to the one I use
2.I think this password scheme is secure
3.I think this password scheme is easy to use
4.I think this password scheme is too hard to memorize
5.I value the ability to create my own password
6.I like this password scheme
7.I think the password is too long
8.I think this password scheme has potential
9.I can see myself using this password scheme

Part 2.5.

Results:
1.~2.5
2.~3
3.~2
4.~4
5.~4.5
6.~2
7.~4.5
8.~1.5
9. ~2

From the results, it seems like the general consensus is that the password scheme is too difficult to memorize. Although it was easy to memorize for the first couple of minutes, asking them again after a 30 minute period was enough to make the participants forget the password entirely. It also seems to have problems with regards to general likability, and users seem to prefer their own passwords. There weren't many people who had good things to say about this password scheme, aside from its ease of use and security. Most users had difficulty memorizing the password, so random numbers may not be the easiest for memorization.
One participant did however, get an easy to memorize password of 55445512, but it was generally difficult for the others to memorize. In fact, some users couldn't even memorize the first half for more than 30 seconds.

Part 2.6.

```
> data3<-read.csv(file.choose(),header=TRUE)
> aggregate(data3[,3:4],list(data3$User),mean)
  Group.1     Time     Login
1    jack 14.50000 0.5000000
2   jamal 40.33333 0.0000000
3     kar 22.33333 0.0000000
4   lokir 12.33333 0.3333333
5  martha 38.00000 0.3333333
6     pan 14.00000 1.0000000
7     rob 15.00000 1.0000000
```

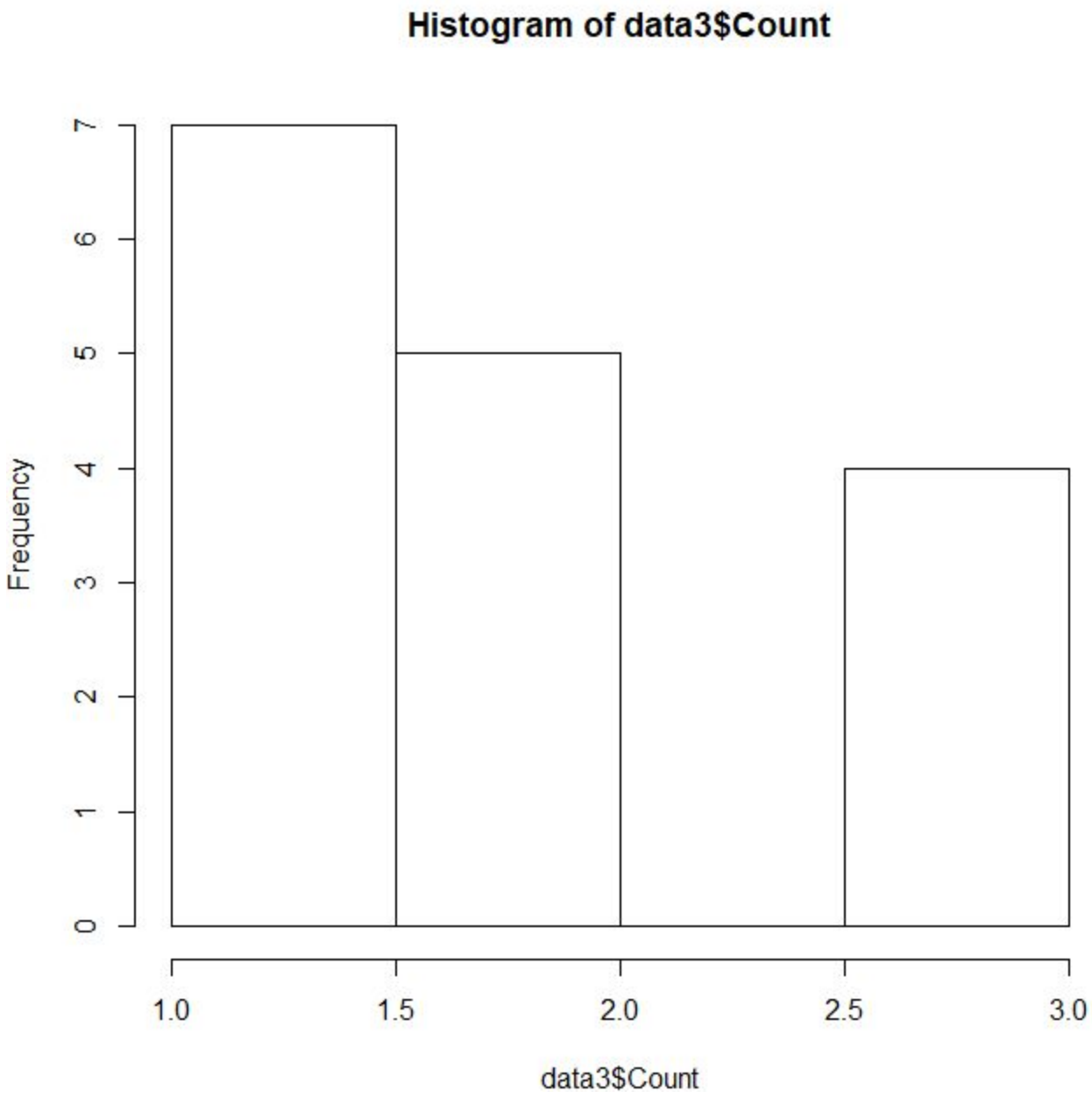
```
> aggregate(data3[,3:4],list(data3$User),median)
  Group.1 Time Login
1    jack 14.5   0.5
2   jamal 37.0   0.0
3     kar 16.0   0.0
4   lokir 11.0   0.0
5  martha 30.0   0.0
6     pan 14.0   1.0
7     rob 15.0   1.0
```

```
> aggregate(data3[,3:4],list(data3$User),sd)
  Group.1      Time     Login
1    jack  6.363961 0.7071068
2   jamal 10.408330 0.0000000
3     kar 20.256686 0.0000000
4   lokir  6.110101 0.5773503
5  martha 19.287302 0.5773503
6     pan        NA        NA
7     rob        NA        NA
```

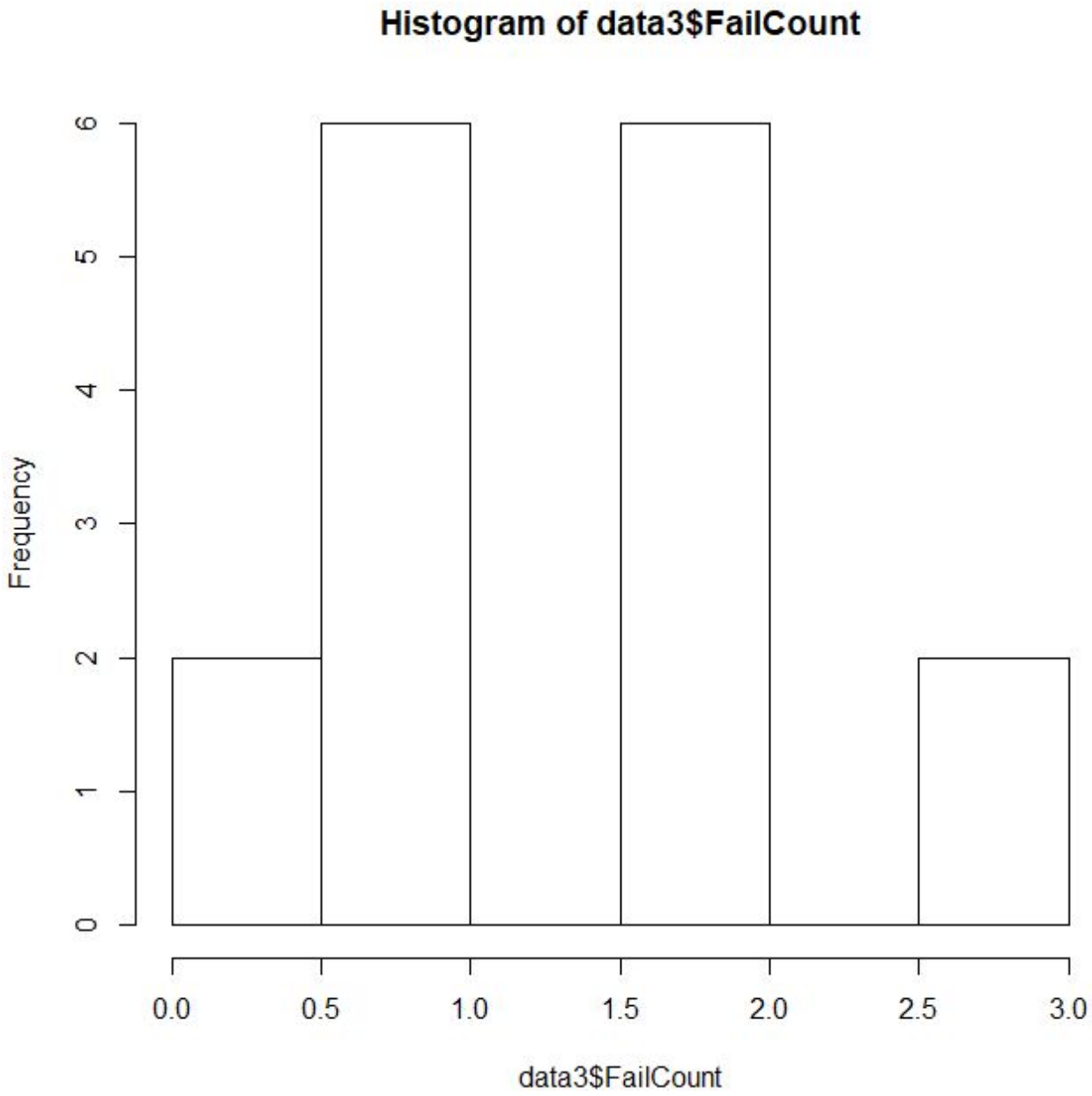
The tables for the login times. N/a means there is no deviation since the user only logged in once.

```
> aggregate(data3[,5:7],list(data3$User),mean)
  Group.1 Count SuccessCount FailCount
1    jack   1.5    0.5000000  1.000000
2   jamal   2.0    0.0000000  2.000000
3     kar   2.0    0.0000000  2.000000
4   lokir   2.0    0.3333333  1.666667
5  martha   2.0    0.3333333  1.666667
6     pan   1.0    1.0000000  0.000000
7     rob   1.0    1.0000000  0.000000
> aggregate(data3[,5:7],list(data3$User),median)
  Group.1 Count SuccessCount FailCount
1    jack   1.5          0.5         1
2   jamal   2.0          0.0         2
3     kar   2.0          0.0         2
4   lokir   2.0          0.0         2
5  martha   2.0          0.0         2
6     pan   1.0          1.0         0
7     rob   1.0          1.0         0
> aggregate(data3[,5:7],list(data3$User),sd)
  Group.1     Count SuccessCount FailCount
1    jack 0.7071068    0.7071068 0.0000000
2   jamal 1.0000000    0.0000000 1.0000000
3     kar 1.0000000    0.0000000 1.0000000
4   lokir 1.0000000    0.5773503 0.5773503
5  martha 1.0000000    0.5773503 0.5773503
6     pan        NA           NA        NA
7     rob        NA           NA        NA
```

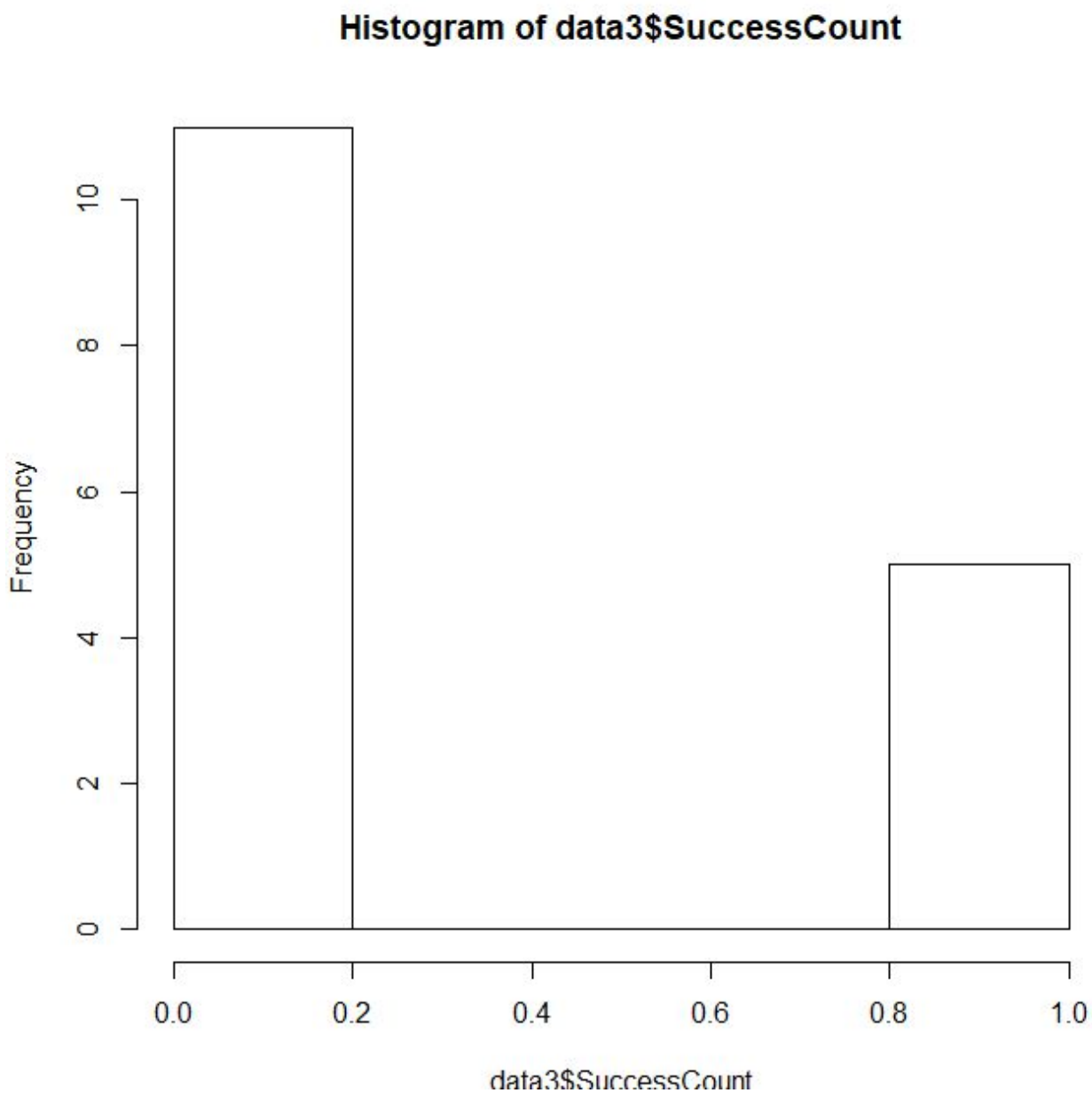
The tables for the login counts. N/a means there is no deviation since the user only logged in once.



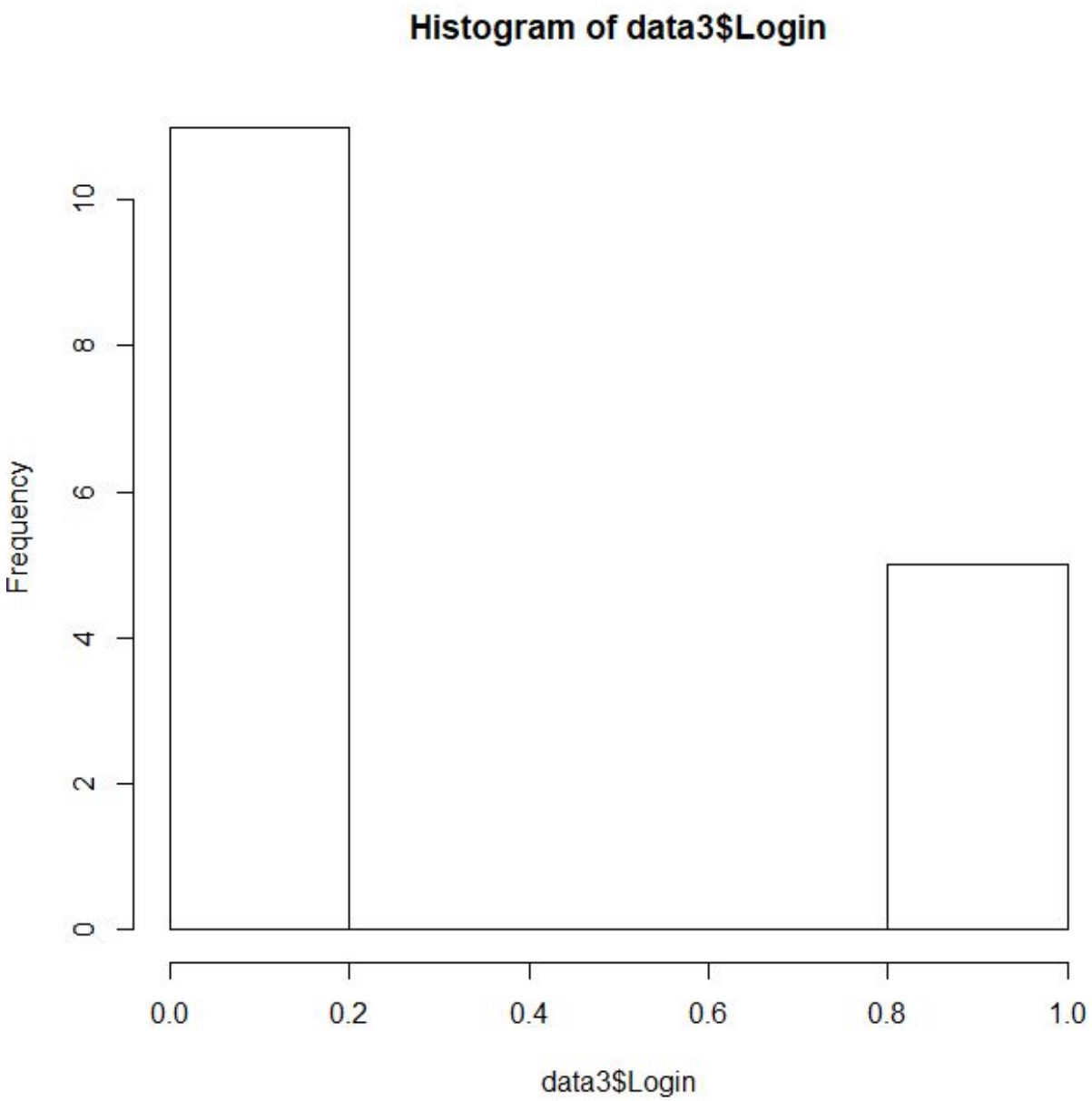
The count of the total number of logins, either successes or fails.



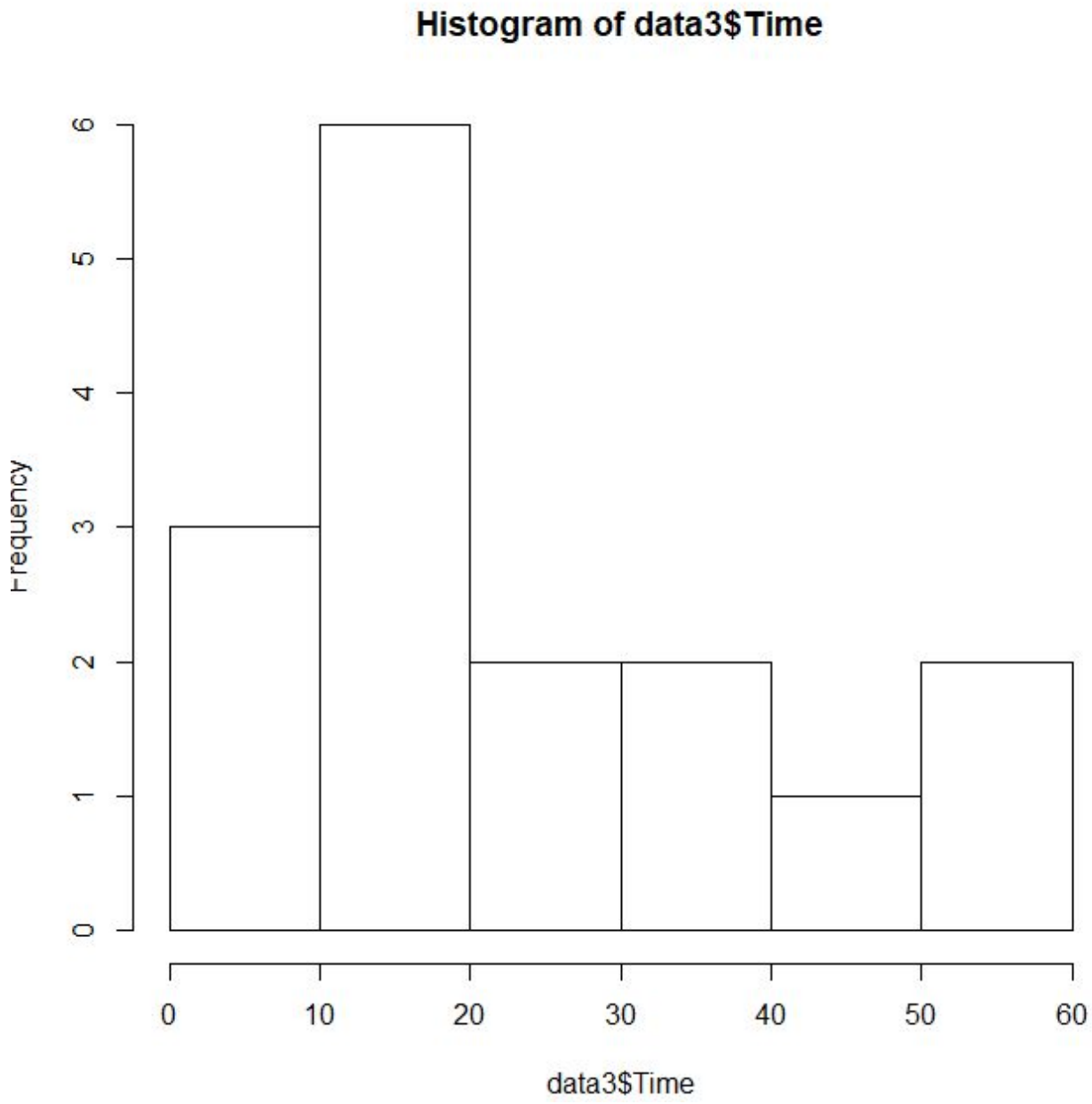
Histogram of the amount of failed logins.



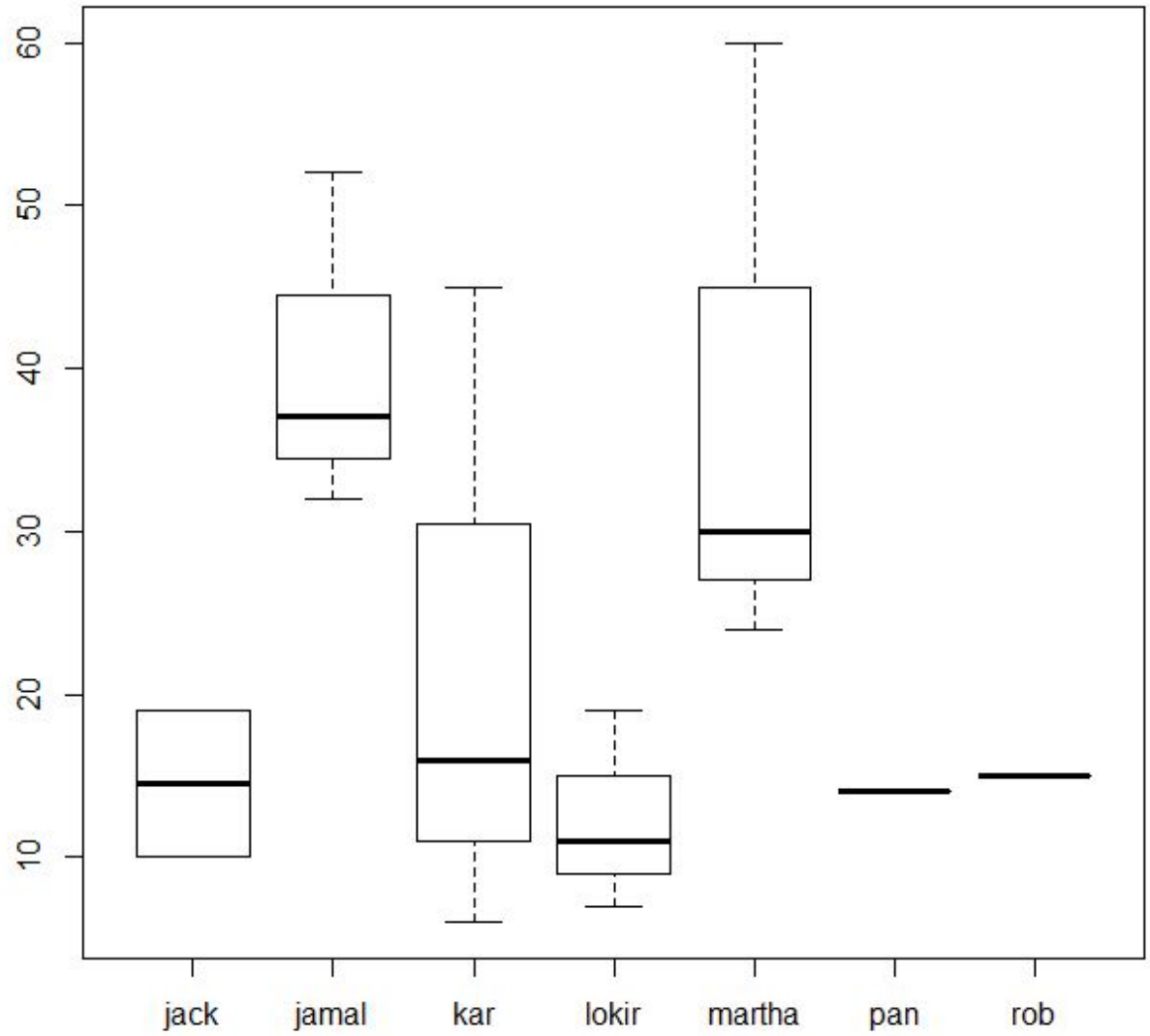
Histogram of the amount of successful logins.



Histogram for the amount of logins. Here there are more failures than successes.



Histogram of the amount of time it took to login.



Boxplot for the amount of time it took to login per user.

Overall, my new password scheme is not superior to that of text21. From my findings, the users were struggling with the password of my scheme (mainly due to its length), while text21's was much easier with 5 characters to memorize rather than 8. It is also much more secure, so there isn't much advantage of my scheme over that of text21's. From these results, I concluded that a wider range of characters (alphabet+0-9), is more advantageous in almost every way than that of a lower range of characters(1-7). The graphs reinforce this with the amount of failures. In conclusion, users would generally not prefer myscheme to a normal password scheme.